RECEIVED
CENTRAL FAX CENTER
DEC 16 2005

Yee & Associates, P.C.

4100 Alpha Road
Suite 1100
Dallas, Texas 75244

Main No. (972) 385-8777
Facsimile (972) 385-7766

FACSIMILE COVER SHEET

To: Commissioner for Patents for **Examiner Syed J. Ali** **Group Art Unit 2195**	Facsimile No. **571/273-8300**
From: Jennifer Pilcher Legal Assistant to Wayne Bailey	No. of Pages Including Cover Sheet: **28**

Enclosed herewith:

- Transmittal document; and
- Appeal Brief.

Re: Application Serial No. **09/735,592**
Attorney Docket No. **AUS9-2000-0720-US1**

Date: Friday, December 16, 2005

Please contact us at (972) 385-8777 if you do not receive all pages indicated above or experience any difficulty in receiving this facsimile.

This Facsimile is intended only for the use of the addressee and, if the addressee is a client or their agent, contains privileged and confidential information. If you are not the intended recipient of this facsimile, you have received this facsimile inadvertently and in error. Any review, dissemination, distribution, or copying is strictly prohibited. If you received this facsimile in error, please notify us by telephone and return the facsimile to us immediately.

PLEASE CONFIRM RECEIPT OF THIS TRANSMISSION BY FAXING A CONFIRMATION TO 972-385-7766.

RECEIVED
CENTRAL FAX CENTER
DEC 16 2005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Brown et al.**

Serial No.: **09/735,592**

Filed: **December 12, 2000**

For: **Language Extension for Light Weight Threading in a JVM**

Group Art Unit: **2195**

Examiner: **Ali, Syed J.**

Attorney Docket No.: **AUS9-2000-0720-US1**

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300 on December 16, 2005.
By: Jennifer Pilcher

12

TRANSMITTAL DOCUMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:
ENCLOSED HEREWITH:

- Appeal Brief (37 C.F.R. 41.37).

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

Respectfully submitted,

Duke W. Yee
Registration No. 34,285
Wayne P. Bailey
Registration No. 34,289
YEE & ASSOCIATES, P.C.
P.O. Box 802333
Dallas, Texas 75380
(972) 385-8777
ATTORNEYS FOR APPLICANTS

RECEIVED
CENTRAL FAX CENTER
DEC 16 2005

Docket No. AUS9-2000-0720-US1

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Brown et al.**

Serial No. **09/735,592**

Filed: **December 12, 2000**

For: **Language Extension for Light Weight Threading in a JVM**

Group Art Unit: **2195**

Examiner: **Ali, Syed J.**

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300 on December 16, 2005.

By: Jennifer Pilcher
Jennifer Pilcher

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on October 17, 2005.

The fees required under § 41.20(B)(2), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

12/19/2005 BABRAHA1 00000012 090447 09735592

01 FC:1402 500.00 DA

PAGE 3/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

PAGE 4/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

PAGE 5/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-22

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: none
2. Claims withdrawn from consideration but not canceled: none
3. Claims pending: 1-22
4. Claims allowed: none
5. Claims rejected: 1-22
6. Claims objected to: none

C. CLAIMS ON APPEAL

The claims on appeal are: 1-22

PAGE 6/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

STATUS OF AMENDMENTS

No amendment after final was filed for this case.

PAGE 7/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

Claim 1 is generally directed to a method for executing code in a multiprocessor environment. When such code is designed, coded or implemented as multi-threaded code, it is possible for such code to be split into multiple portions or threads that allow for concurrently operation of the threads to take advantage of the plurality of processors that exist in such multiprocessor environment. In such an environment, where multiple code portions or threads are concurrently executing, there is a need to coordinate or synchronize such plurality of executions so that the overall program code achieves its desired results. However, in present systems there is an underlying tension between hardware designers and software designers as to how to best achieve such code coordination/synchronization, where the underlying hardware may be designed in such a way that the intentions of the software programmer/developer gets lost in the execution of the code. In the prior art, software could only take advantage of multiple processors in a process if multiple threads were used. By default, it was assumed that all statements were executed in the order written, and further, that whatever the effect of a given statement, the execution was finished before the beginning of the next statement. The present invention provides a mechanism for indicating that a statement, block of code, or method is safe to execute out of order by using special in-line keywords that indicate that a subsequent statement/block-of-code/method may be executed out of order. Specifically, Claim 1 is directed to a method for asynchronous (i.e. out of order) execution within a program. Code is executed in a first thread. A determination is made as to whether a first keyword exists in this code, *the first keyword being a flag indicating that a subsequent code element following the first keyword may be executed out of order.* This subsequent code element is executed in a second thread, thereby advantageously providing the desired out-of-order code execution as determined during program run-time (Specification page 11, second paragraph – page 15, bottom of page).

B. CLAIM 5 - INDEPENDENT

Claim 5 is directed to a method for asynchronous execution within a program. Code is executed in a first thread. A determination is made as to whether a first keyword exists in this

PAGE 8/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

code, *the first keyword being a flag indicating that a subsequent code element following the first keyword may be executed out of order.* This subsequent code element is executed in a second thread. Further, a determination is made as to whether a second keyword exists in the code, the *second keyword indicating that execution of the code element in the second thread must complete before a next code element immediately following the second keyword is executed.* The next code element is executed in the first thread after execution of the code element in the second thread completes, thereby advantageously providing the desired multi-threaded run-time synchronization (Specification page 11, second paragraph – page 15, bottom of page).

C. CLAIM 10 – INDEPENDENT AND MEANS-PLUS-FUNCTION

Claim 10 is directed to an apparatus for asynchronous execution within a program. Code is executed, by the apparatus, in a first thread. A determination is made, by the apparatus, as to whether a first keyword exists in this code, *the first keyword being a type definition indicating that a subsequent code element following the first keyword may be executed out of order.* This subsequent code element is executed, by the apparatus, in a second thread to thereby advantageously providing the desired out-of-order code execution as determined during program run-time (Specification page 11, second paragraph – page 15, bottom of page).

In the preferred embodiment, the structure, material or acts corresponding to the first means, second means, third means and fourth means is described at Specification page 5, first paragraph and extending to page 14, first paragraph, and depicted in Figures 1-4 (all blocks).

D. CLAIM 14 - INDEPENDENT AND MEANS-PLUS-FUNCTION

Claim 14 is directed to an apparatus claim of similar scope to Claim 5, and the summary of Claim 5 given above is equally applicable to Claim 14, and is thus hereby incorporated by reference in order to provide the summary of Claim 14.

In the preferred embodiment, the structure, material or acts corresponding to the first means, second means, third means and fourth means is described at Specification page 5, first paragraph and extending to page 14, first paragraph, and depicted in Figures 1-4 (all blocks).

E. CLAIM 17 - INDEPENDENT

Claim 17 is directed to an apparatus for asynchronous execution within a program. The apparatus includes both an interpreter and a program, the program including a first keyword

PAGE 9/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

indicating a code element that may be executed out of order. The interpreter, upon detecting the keyword, creates a light weight thread and executes the code element in the light weight thread, thereby advantageously providing the desired out-of-order code execution as determined during program run-time (Specification page 11, second paragraph – page 15, bottom of page).

F. CLAIM 19 - INDEPENDENT

Claim 19 is directed to a computer program product claim of similar scope to Claim 1, and the summary of Claim 1 given above is equally applicable to Claim 19, and is thus hereby incorporated by reference in order to provide the summary of Claim 19.

G. CLAIM 21 - INDEPENDENT

Claim 21 is a computer program product claim of similar scope to Claim 5, and the summary of Claim 5 given above is equally applicable to Claim 21, and is thus hereby incorporated by reference in order to provide the summary of Claim 21.

PAGE 10/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. GROUND OF REJECTION 1 (Claims 10-18)

Claims 10-18 stand rejected under 35 U.S.C. § 101 as being directed to non-statutory subject matter.

B. GROUND OF REJECTION 2 (Claims 1-22)

Claims 1-22 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Bonachea ("Bulk File I/O Extensions to Java").

PAGE 11/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 10-18)

A.1. Claims 10-18

With respect to Claim 10, Appellants previously amended Claims 10, 14 and 15 to explicitly recite 'apparatus' in the body of the claim. It is urged that such an apparatus comprising the specifically enumerated elements is statutory subject matter under 35 U.S.C. § 101, and thus Claims 10-16 are shown to have been erroneously rejected under 35 U.S.C. § 101.

Further, Claims 10 (and dependent Claims 11-13 and 16) and 14 (and dependent Claim 15) are means-plus-function claims. Per 35 USC 112, 6th paragraph:

> An element in a claim for a combination may be expressed as a means or step for performing a specified function *without the recital of structure*, material, or acts in support thereof, and *such claim shall be construed to cover the corresponding structure*, material, or acts described in the specification and equivalents thereof (emphasis added by Appellants).

It is thus urged that the Examiner has failed to comply with the requirement of 35 USC 112, 6th paragraph by failing to construe such claims as covering the corresponding structure described in the specification (and equivalents thereof), and thus has erroneously rejected such claims as being directed to non-statutory subject matter.

Still further with respect to Claim 13, such claim specifically recites that the first thread is executed on a *first processor* and the second thread is executed on a *second processor* – further evidencing that the claimed subject matter is statutory under 35 U.S.C. § 101.

With respect to Claim 17 (and similarly for dependent Claim 18), such claim recites "An apparatus for asynchronous execution within a program". Per M.P.E.P 2106 (II)(A):

> **Only when the claim is devoid of any limitation to a practical application in the technological arts should it be rejected under 35 U.S.C. 101.** Compare *Musgrave*, 431 F.2d at 893, 167 USPQ at 289; *In re Foster*, 438 F.2d 1011, 1013, 169 USPQ 99, 101 (CCPA 1971) (emphasis added by Appellants).

PAGE 12/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

Because Claim 17 is *not* devoid of any limitation to a practical application in the technological arts, but instead specifically recites an apparatus for asynchronous execution within a program (including the specifically claimed element of an interpreter that, upon detecting a keyword, creates a light weight thread and executes the code element in the light weight thread), it is thus urged that such claimed apparatus is statutory subject matter under 35 U.S.C. § 101 as having a practical application in the technological arts. Thus, Claims 17 and 18 are shown to have been erroneously rejected under 35 U.S.C. § 101.

B. GROUND OF REJECTION 2 (Claims 1-22)

B.1. Claims 1, 2, 9 and 19

The cited reference teaches *a series of asynchronous libraries that are called* to perform asynchronous I/O operations. The present invention is instead directed to a simpler and more elegant approach to enabling asynchronous operations to occur, through *use of inline keywords* that flag or direct that subsequent code be processed in a particular fashion during code execution – i.e. *a runtime determination*. The cited reference does not teach or otherwise suggest providing any such runtime determination capability, as the reference describes use of two compilers, a Titanium compiler and a C compiler (Section 2.2, second paragraph) which compile code and this code is then linked using a special library into executable code that is subsequently executed. *There is no abillty to detect asynchronous flags during code execution.* The cited reference could not even be modified in accordance with Claim 1 without essentially gutting the essence, expressed purpose, and stated objectives of the teachings of the cited reference (a two-step compile, followed by a linking to special libraries to achieve high performance). The fact that a prior art device could be modified so as to produce the claimed device is not a basis for an obviousness rejection unless the prior art suggested the desirability of such a modification. *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984). Although a device may be capable of being modified to run the way [the patent applicant's] apparatus is claimed, there must be a suggestion or motivation *in the reference* to do so. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). Because the overall architecture and implementation details are substantially different – the cited reference teaching a traditional compile/link/execute process, where any code particulars are determined during linkage and therefore are static in nature – there would have been no suggestion or other motivation to modify the teachings contained therein to provide

PAGE 13/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

a dynamic, run-time determination as to whether to execute an in-line code element in another thread based upon whether an in-line flag is encountered. Thus, it is urged that Claim 1 has been erroneously rejected under 35 USC 103(a), as a proper prima facie showing of obviousness has not been established by the Examiner[1].

B.2. Claim 3

Appellants initially show error in the rejection of Claim 3 for similar reasons to those given above with respect to Claim 1.

Further with respect to Claim 3, such claim includes features that further emphasize the flexibility provided by the present invention. The first keyword is usable both within a method, as shown at 404 of Figure 4A, as well as a type definition for a method, as shown at 414 of Figure 4B. This flexibility in use advantageously provides that a block of code can include statements that are executed asynchronously with respect to the nesting level of that block, or alternatively to provide asynchronous processing at the block level if resource constrained (Specification page 12, last paragraph which extends to page 13). The cited reference does not teach or suggest either this claimed feature or its resulting advantages. Thus, it is further urged that Claim 3 has been erroneously rejected, as there are missing claimed features not taught or suggested by the cited reference.

B.3. Claims 4 and 20

Appellants initially show error in the rejection of Claim 4 (and similarly for Claim 20) for similar reasons to those given above with respect to Claim 1.

Further with respect to Claim 4 (and similarly for Claim 20), Appellants urge that the cited reference does not teach or suggest the claimed feature of "wherein the first thread is executed on a first processor and the second thread is executed on a second processor". As can be seen, the first thread that is executing is executed on a first processor, and a determination is

[1] In rejecting claims under 35 U.S.C. Section 103, the examiner bears the initial burden of presenting a prima facie case of obviousness. *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992). Only if that burden is met, does the burden of coming forward with evidence or argument shift to the applicant. *Id.* To establish prima facie obviousness of a claimed invention, all of the claim limitations must be taught or suggested by the prior art. MPEP 2143.03 (emphasis added by Appellants). *See also, In re Royka*, 490 F.2d 580 (C.C.P.A. 1974). If the examiner fails to establish a prima facie case, the rejection is improper and will be overturned. *In re Fine*, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988).

PAGE 14/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

made that a subsequent code element can be executed out of order (per independent Claim 1). This code element is then executed in a second thread on another processor (Claim 1 in combination with Claim 4). The cited reference does not teach or suggest any ability to spawn, create, or otherwise invoke a second thread in another processor during code execution in a first processor, and there would have been no motivation to modify the teachings contained therein to include such missing claimed feature due to inherent latencies that would be introduced by such action (and such latencies being the exact thing that the teachings of the cited reference are attempting to eliminate in their high-performance design). While the cited reference alludes to grid-computing, such high-level statement does not teach or otherwise suggest the specific steps cited in Claim 4 (in combination with Claim 1). In all likelihood (although the reference does not describe any details of this grid-computing), decisions regarding what code is executed on what processor is determined at compile time, and a plurality of independent executables are generated on a per-processor basis to be executed independent of one another. Such a configuration does not teach or otherwise suggest a first thread is executing on a first processor, and a determination is made during such execution that a subsequent code element can be executed out of order (per independent Claim 1), with this code element then being executed in a second thread on another processor. Thus, it is urged that Claim 4 has been erroneously rejected under 35 USC 103(a) as there are missing claimed features not taught or suggested by the cited reference.

B.4. Claims 5, 14 and 21

Appellants initially show error in the rejection of Claim 5 (and similarly for Claims 14 and 21) for similar reasons to those given above with respect to Claim 1.

Further with respect to Claim 5 (and similarly for Claims 14 and 21), such claim recites both a first keyword and a second keyword. The first keyword indicates a code element that may be executed out of order. The second keyword indicates that execution of the code element in the second thread (as executed by the encountered the first keyword) must complete *before the next code element is executed.* Thus, these two keywords synergistically co-act to provide an ability to resynchronize code execution without polling for completion, as was done in the past. In fact, that is exactly how the cited reference achieves synchronization – by polling. As can be seen at Section 4.1.3 of the cited reference, "Done" methods are defined to achieve

PAGE 15/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

synchronization, and these "Done" methods check the status of the AsyncFileRequests the application is querying and returns a Boolean flag indicating whether the "Done" condition was satisfied. Thus, this "Done" routine is a polling routine that is used to determine if a previously dispatched task has completed – in other words, it is looking at the status of *past actions*. In contrast, Claim 5 is directed to a *forward-looking methodology*, where the second keyword indicates that execution of the code element in the second thread must complete *before the next code element is executed*. It is, in effect, a brake that is applied on the next code element as can be seen by the delayed 'stuff2' execution shown in Figure 4E. The teachings of the cited reference do not teach or otherwise suggest any type of keyword the directly effects code execution of subsequent code, but instead merely provides a polling method to determine whether a previously dispatched task has completed. Thus, it is shown that Claim 5 has been erroneously rejected under 35 USC 103(a), as there are missing claimed elements not taught/suggested by the cited reference.

Still further with respect to Claim 5, since the cited reference teaches a simple polling technique that merely looks back at previously dispatched tasks, and is not forward looking or otherwise forward-deterministic, it is shown that the cited reference does not teach the claimed step of "executing the next code element *in the first thread* after execution of the code element *in the second thread* completes" as there is no type of thread coordination provided by the simple "Done" polling technique that is provided by the teachings of the cited reference. Thus, Claim 5 is further shown to have been erroneously rejected under 35 USC 103(a), as there are additional missing claimed elements not taught/suggested by the cited reference.

B.5. Claims 6, 15 and 23

Appellants initially show error in the rejection of Claim 6 (and similarly for Claims 15 and 24) for similar reasons to those given above with respect to Claims 1 and 5.

Further with respect to Claim 6 (and similarly for Claims 15 and 24), Appellants show that such claim recites a feature of "determining whether a third keyword *exists in the code element*, the third keyword indicating a statement that may be executed out of order". Thus, the code element for which out of order execution has been detected (by the first keyword) has, itself, a keyword that indicates a statement may be executed out-of-order, thus advantageously providing *a nesting capability for out-of-order execution*. The cited reference does not teach or

PAGE 16/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

otherwise suggest any such nested out-of-order execution. This nesting capability advantageously provides that a block of code can include statements that are executed asynchronously with respect to the nesting level of that block, as well as having other blocks of code nested within such block of code, to thereby provide recursive asynchronous execution of such block of code (Specification page 12, last paragraph). The cited reference does not teach/suggest either this claimed feature or its resulting advantage. Therefore, it is further urged that Claim 6 is not obvious in view of the cited reference as there are missing claimed features not taught/suggested by such cited reference.

B.6. Claim 7

Appellants initially show error in the rejection of Claim 7 for similar reasons to those given above with respect to Claim 1.

Further with respect to Claim 7, Appellants urge that the cited reference does not teach or suggest the claimed feature of "wherein the method is executed by an interpreter". Rather, the cited reference teaches use of two compilers – a Titanium compiler and a C compiler – where code is compiled and then linked using a special library to achieve its stated performance objectives. Because of this compiler processing methodology, there is no ability to determine, *during code execution*, whether a first keyword exists in the code, the first keyword being a flag indicating that a subsequent code element following the first keyword may be executed out of order. Therefore, it is further urged that Claim 7 is not obvious in view of the cited reference as there are missing claimed features not taught/suggested by such cited reference.

B.7. Claim 8

Appellants initially show error in the rejection of Claim 8 for similar reasons to those given above with respect to Claim 7.

Further, Appellants urge that the fact that a prior art device could be modified so as to produce the claimed device is not a basis for an obviousness rejection unless the prior art suggested the desirability of such a modification. *In re Gordon, supra.* The cited reference teaches a technique for reducing latencies in I/O operations – and hence is directed to a technique for making software run faster in order to enable high-performance scientific applications (Section 1, Introduction; Section 2.2, Titanium). For example, the cite reference explicitly states:

PAGE 17/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

"This research focuses on maximizing the I/O performance for each node given a fixed workload." (emphasis added by Appellants)

The cited reference teaches use of two different compilers and associated compiling of code to achieve its performance objectives – the Titanium compiler performs various optimizations using knowledge of the parallel flow control, and then translates programs entirely to C where they are further optimized during a second compile operation (Section 2.2, second paragraph). Modifying the teachings of the cited reference in accordance with Claim 8 would in fact result in these teachings *failing to meet their stated objectives*, as a Java Virtual Machine interpreter is intrinsically slower in operation than the environment as taught by the cited reference. In addition, the current state of such JVMs do not provide optimization using knowledge of parallel flow control – an expressed required feature of the teachings of the cited reference. Thus, a person of ordinary skill in the art would not have been motivated to modify the teachings of the cited reference in accordance with the teachings of the claimed invention as the expressed desires and purposes of the cited reference would have been unachievable – strongly evidencing no motivation to modify such teachings in accordance with Claim 8. It is therefore shown that there was no suggestion of any desire to modify the teachings of the cited reference to include a Java Virtual Machine interpreter, and thus the basis of this obviousness rejection is shown to be error, per *In re Gordon, Id.*

Further, because there was no suggestion of any desire to modify the teachings of the cited reference in accordance with Claim 8, the only motivation for such modification must be coming from Appellants' own patent specification, which is improper hindsight analysis. It is error to reconstruct the patentee's claimed invention from the prior art by using the patentee's claims as a "blueprint". *Interconnect Planning Corp. v. Feil*, 774 F.2d 1132, 227 USPQ 543 (Fed. Cir. 1985). Thus, Claim 8 is still further shown to have been erroneously rejected using impermissible hindsight analysis.

This failure by the reference to provide any suggestion to modify the teachings therein in accordance with Claim 8 can further be seen by the fact that the cited reference *expressly teaches away* from using an interpreter such as a Java Virtual Machine (JVM) due to resulting performance inadequacies (Section 2.2). This further evidences improper hindsight analysis being used by the Examiner in rejecting Claim 8, as the only motivation for making this

PAGE 18/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

modification in rejecting Claim 8 must be coming from Appellants' own patent specification, which is improper hindsight analysis. Thus, Claim 8 is further shown to have been erroneously rejected under 35 USC 103(a).

B.8. Claims 10, 11 and 16

Appellants initially show error in the rejection of Claim 10 (and similarly for Claims 11 and 16) for similar reasons to those given above with respect to Claim 1.

Further with respect to Claim 10 (and similarly for Claims 11 and 16), such claim recites that the *first keyword*, which is used to indicate an out-of-order capability, *is a type definition for a code element*, which advantageously allows for changing the actual declaration of a function itself, as shown in Applicants' preferred embodiment in Figure 4B and described in the Specification at page 11, last paragraph. In contrast, per the teachings of the cited reference, a special purpose library is provided having unique names for individual methods that are invoked to perform asynchronous I/O operations. Thus, it is urged that the features of Claim 10 are not taught or suggested by the cited reference, and therefore is it shown that Claim 10 has been erroneously rejected under 35 USC 103(a).

B.9. Claim 12

Appellants initially show error in the rejection of Claim 12 for similar reasons to those given above with respect to Claim 10.

Further with respect to Claim 12, such claim includes features that further emphasize the flexibility provided by the present invention. The first keyword is usable both within a method, as shown at 404 of Figure 4A, as well as a type definition for a method, as shown at 414 of Figure 4B. This flexibility in use advantageously provides that a block of code can include statements that are executed asynchronously with respect to the nesting level of that block, or alternatively to provide asynchronous processing at the block level if resource constrained (Specification page 12, last paragraph which extends to page 13). The cited reference does not teach or suggest either this claimed feature or its resulting advantage. Thus, it is further urged that Claim 3 has been erroneously rejected, as there are missing claimed features not taught or suggested by the cited reference.

PAGE 19/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

B.10. Claim 13

Appellants initially show error in the rejection of Claim 13 for similar reasons to those given above with respect to Claim 10.

Appellants further show error in the rejection of Claim 13 for similar reasons to those given above with respect to Claim 4.

B.11. Claim 17

With respect to Claim 17, such claim recites the claimed feature of "wherein the interpreter, upon detecting the keyword, creates a light weight thread and executes the code element in the light weight thread". As can be seen, a light weight thread is *created by the interpreter* upon detecting the keyword, and the code element executes in the light weight thread. The cited reference does not teach an interpreter that creates a light weight thread upon detecting a keyword, as expressly recited in Claim 17. Because of the substantial architectural differences between the teachings of the cited reference and the invention recited in Claim 17, there is no ability of an interpreter to create a light weight thread upon detecting a keyword, and then executing code in the created thread due to use of a compiler/linker that generates machine executable code that is subsequently executed by a processor. In rejecting Claim 17, the Examiner asserts that the cited reference teaches this claimed feature of a thread creation by an interpreter at section 4.1.2. Appellants urge that this cited passage is directed to details of I/O initiation methods, and when these methods are called, they perform type-checking, bounds-checking and end-of-file checking, and then initiate the requested asynchronous I/O operation. There is no indication that this asynchronous I/O operation is executed in a newly created thread that is created by an interpreter itself. Thus, it is urged that Claim 17 has been erroneously rejected under 35 USC 103(a) as there are missing claimed features not taught or suggested by the cited reference.

B.12. Claim 18

Appellants initially show error in the rejection of Claim 18 for similar reasons to those given above with respect to Claim 17.

Appellants further show error in the rejection of Claim 18 for similar reasons to those given above with respect to Claim 8.

PAGE 20/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

In conclusion, Appellants have shown numerous errors in the Examiner's rejection of all pending claims, and respectfully requests that this Board reverse all such rejections.

Duke W. Yee
Reg. No. 34,285
Wayne P. Bailey
Reg. No. 34,289
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

PAGE 21/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A method for asynchronous execution within a program, comprising:

executing code in a first thread;

during the executing of the code, determining whether a first keyword exists in the code, the first keyword being a flag indicating that a subsequent code element following the first keyword may be executed out of order; and

executing the code element in a second thread.

2. The method of claim 1, wherein the code element is one of an instruction, a block, and a method.

3. The method of claim 1, wherein the first keyword is usable in both an internal definition of a method and a type definition for the method.

4. The method of claim 1, wherein the first thread is executed on a first processor and the second thread is executed on a second processor.

5. A method for asynchronous execution within a program, comprising:

executing code in a first thread;

determining whether a first keyword exists in the code, the first keyword indicating a code element that may be executed out of order;

PAGE 22/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

executing the code element in a second thread;

determining whether a second keyword exists in the code, the second keyword indicating that execution of the code element in the second thread must complete before a next code element immediately following the second keyword is executed; and

executing the next code element in the first thread after execution of the code element in the second thread completes.

6. The method of claim 5, further comprising:

determining whether a third keyword exists in the code element, the third keyword indicating a statement that may be executed out of order; and

executing the statement in a third thread.

7. The method of claim 1, wherein the method is executed by an interpreter.

8. The method of claim 7, wherein the interpreter is a Java virtual machine.

9. The method of claim 1, wherein the second thread is a light weight thread.

10. An apparatus for asynchronous execution within a program, comprising:

first execution means for executing, by the apparatus, code in a first thread;

determination means for determining, by the apparatus, whether a first keyword exists in the code, the first keyword being a type definition indicating that a subsequent code element following the first keyword may be executed out of order; and

PAGE 23/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

second execution means for executing, by the apparatus, the code element in a second thread.

11. The apparatus of claim 10, wherein the code element is one of an instruction, a block, and a method.

12. The apparatus of claim 10, wherein the first keyword is usable in both an internal definition of a method and a type definition for the method.

13. The apparatus of claim 10, wherein the first thread is executed on a first processor and the second thread is executed on a second processor.

14. An apparatus for asynchronous execution within a program, comprising:

first execution means for executing, by the apparatus, code in a first thread;

determination means for determining, by the apparatus, whether a first keyword exists in the code, the first keyword indicating a code element that may be executed out of order;

second execution means for executing, by the apparatus, the code element in a second thread;

means for determining, by the apparatus, whether a second keyword exists in the code, the second keyword indicating that execution of the code element in the second thread must complete before a next code element immediately following the second keyword is executed; and

means for executing, by the apparatus, the next code element in the first thread after execution of the code element in the second thread completes.

PAGE 24/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

15. The apparatus of claim 14, further comprising:

means for determining, by the apparatus, whether a third keyword exists in the code element, the third keyword indicating a statement that may be executed out of order; and

means for executing, by the apparatus, the statement in a third thread.

16. The apparatus of claim 10, wherein the second thread is a light weight thread.

17. An apparatus for asynchronous execution within a program, comprising:

an interpreter; and

a program, the program including a first keyword indicating a code element that may be executed out of order,

wherein the interpreter, upon detecting the keyword, creates a light weight thread and executes the code element in the light weight thread.

18. The apparatus of claim 17, wherein the interpreter is a Java virtual machine.

19. A computer program product, in a computer readable medium, for asynchronous execution within a program, comprising:

instructions for executing code in a first thread;

instructions for determining, during the executing of the code, whether a first keyword exists in the code, the first keyword being a flag indicating that a subsequent code element following the first keyword may be executed out of order; and

instructions for executing the code element in a second thread.

PAGE 25/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

20. The computer program product of claim 19, wherein the first thread is executed on a first processor and the second thread is executed on a second processor.

21. A computer program product, in a computer readable medium, for asynchronous execution within a program, comprising:

instructions for executing code in a first thread;

instructions for determining whether a first keyword exists in the code, the first keyword indicating a code element that may be executed out of order;

instructions for executing the code element in a second thread;

instructions for determining whether a second keyword exists in the code, the second keyword indicating that execution of the code element in the second thread must complete before a next code element immediately following the second keyword is executed; and

instructions for executing the next code element in the first thread after execution of the code element in the second thread completes.

22. The computer program product of claim 21, further comprising:

instructions for determining whether a third keyword exists in the code element, the third keyword indicating a statement that may be executed out of order; and

instructions for executing the statement in a third thread.

PAGE 26/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

EVIDENCE APPENDIX

There is no evidence to be presented.

PAGE 27/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.

PAGE 28/28 * RCVD AT 12/16/2005 10:20:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/24 * DNIS:2738300 * CSID:972 385 7766 * DURATION (mm-ss):06-06